[SIGIR/SIRIP] 김민석 실험 일지 REFINED

tldr: 예측 → 랭킹 → 랭킹한것들 예측 → 메타모델 →성능향상이 있는지 확인

Ranking based voting 인데 살짝 다르게 접근해볼 생각입니다.

현재 샘플에 대해서 long, short, hold인지 예측을 합니다.

이후에 현재 샘플과 유사한 샘플들 top K개를 랭킹해서 해당 유사 샘플들에 대해서 어떤식으로 모델이 예측을 하는지, 그리고 실제로는 레이블이 뭐였는지를 고려해서 현재 샘플에 대한 예측을 더 좋은 방향으로 수정 가능한지 확인해봅니다.

여기서 랭킹하는 방법과 예측에 사용되는 모델을 vary하면서 실험해봅니다.

CBITS 스타일의 예측 (three class classification)

(1) 예측에 활용되는 모델: XGBoost, Transformer (더 추가 가능: TFT, TabTransformer, TabNet, LSTM등)

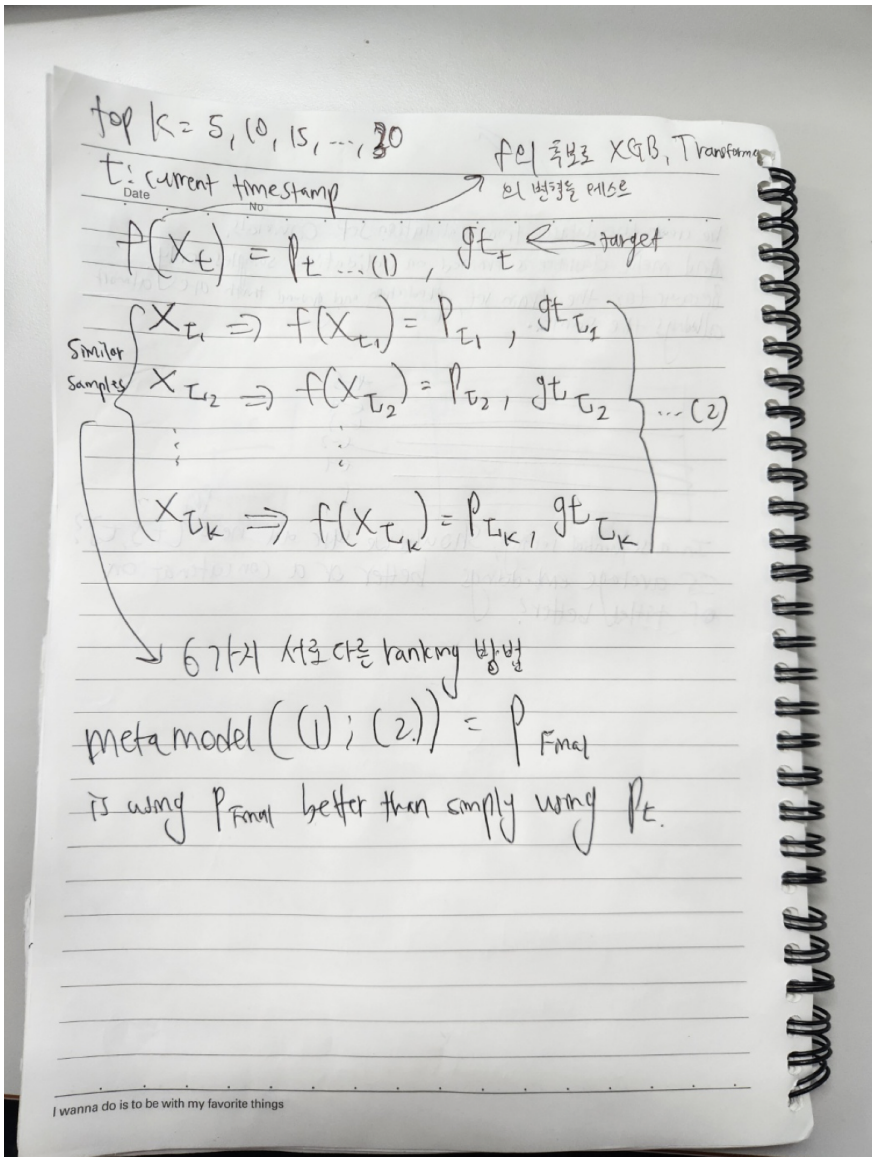
- 시간이랑 short paper라는 점을 고려해서 XGBoost랑 Transformer 두개만 하는게 충분할 것 같다. 그리고 Transformer의 경우 조금 robust 하게 만들어볼 생각인데, TS2Vec 도입, DAIN 혹은 RevIN 방법의 input normalization 그리고 news embedding 과 news sentiment score를 둘 다 활용해서 multimodal 한 버전으로 진행도 가능할듯. Loss도 weighted cross entropy, focal loss 둘 다 사용 가능하다 (하나만 정해서 하는게 좋을듯). 간단하게 하려면 그냥 weighted cross entropy를 사용하는게 좋을듯하다.
- XGBoost, XGBoost + Sentiment Scores, Transformer, Transformer + Sentiment Scores, Transformer + News Embeddings

(2) 랭킹에 사용되는 방법들: news sentiment score, news embedding method, chart Euclidean Distance, chart DTW, TS2Vec, UMAP(news embedding) + TS2Vec

(3) 랭킹된 유사 샘플 + 현재 샘플에 대한 예측들을 (1) 모델을 활용해서 얻어내고, 랭킹된 유사 샘플의 예측 레이블, 랭킹된 유사 샘플들에 대한 ground truth label, 현재 샘플에 대한 예측 레이블을 다같이 meta model (4번)에 입력으로 집어넣는다.

(4) 메타모델: KNeighborsClassifier, SVC, GaussianProcessClassifier, DecisionTreeClassifier, RandomForestClassifier, MLPClassifier, AdaBoostClassifier, GaussianNB, QuadraticDiscriminantAnalysis, LogisticRegression 등 sklearn에 있는 classifier 들을 전부 활용해서 실험해본다. 이 모델들끼리 서로 큰 차이가 있을지는 아직 불명.

- 추가로 catboost, tabnet, tabtransformer 도 실험 가능.
- categorical data에 대해서 robust한 성능을 가진 tabular model이면 될듯

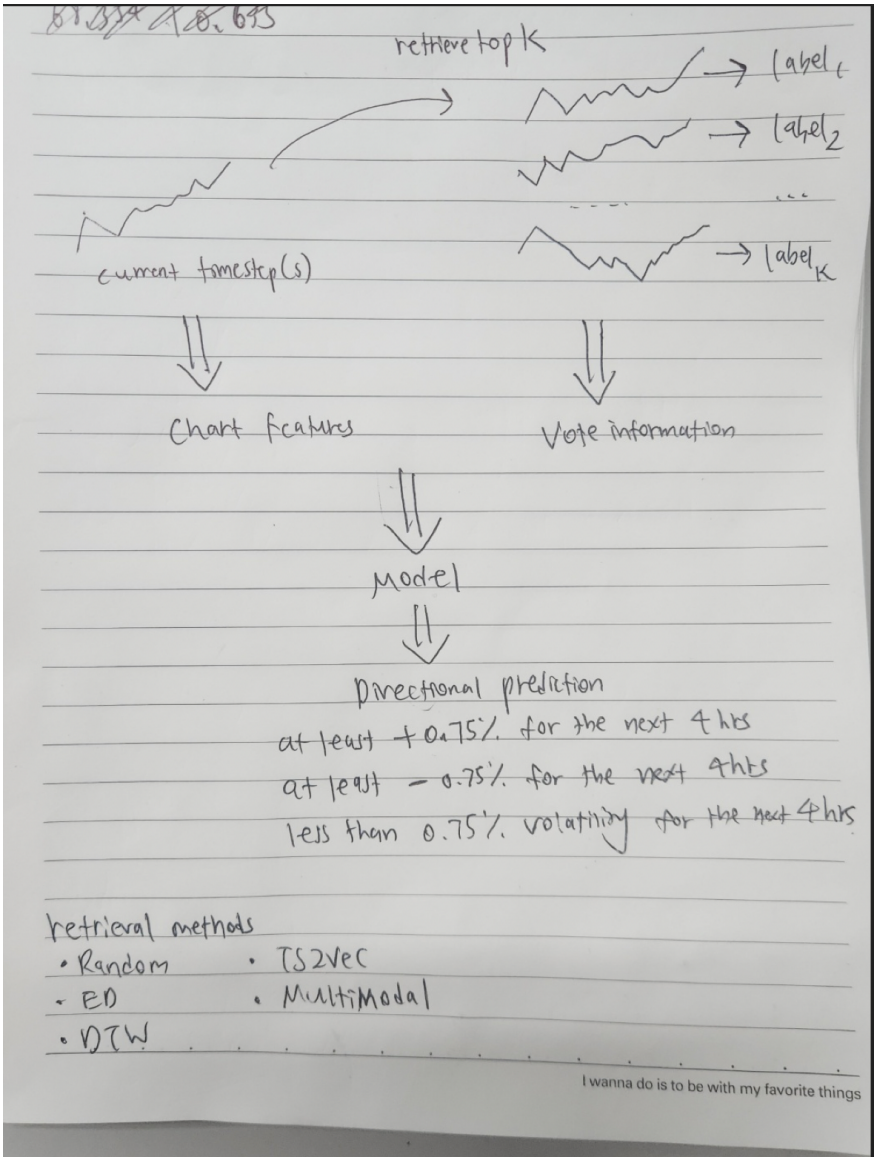


***** 결론적으로 위 아이디어는 실패. 성능향상은 없고 오히려 성능 저하를 불러일으킨다 *****

그래서 다른 방식으로 진행하고 있다.

랭킹 후 voting을 한뒤에 voting 정보를 최종 모델의 추가 입력 피처로 사용한다.

Raw Voting score를 사용하는 방법과 Softmax normalize 된 voting score를 사용하는 방법 두가지가 있는데, 현재 엄청나게 많은 실험을 해본건 아니지만 후자가 더 좋아보인다. Normalization을 통해서 다른 피처 값들과 비슷한 range를 형성하기 때문에 전체적으로 학습할때 더 stable한듯.



실험 셋팅: BTC-USDT 4시간봉

Candidate pool: 9449개의 timestep

Train data: 1890개의 timestep

Validation data: 236개의 timestep

Test data: 237개의 timestep

transformer lookback window length = 6

80% candidate pool, 20% train/val/test/split

Naive Baselines

[코드]

	Random Agent
Accuracy (%)	33.434

Weighted F1	0.393

Random Sampling Based Ranking

[랜덤 기반 과거 데이터 수집 코드]

XGBoost [코드] [코드 100회 반복]

첫번째는 10회 반복, 두번째는 100회 반복

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	51.899	53.080	53.671	54.641	54.599	54.008	**55.021**
Weighted F1	0.580	0.589	0.592	0.603	0.602	0.597	**0.605**

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	51.899	**54.084**	53.873	53.608	53.485	53.840	53.658
Weighted F1	0.580	**0.598**	0.595	0.593	0.592	0.595	0.593

Transformer [코드] [코드 100회 반복]

10회 반복

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	68.354	62.236	64.473	65.738	66.751	67.300	**68.017**
Weighted F1	0.635	0.617	0.617	0.634	0.636	**0.638**	0.637

Euclidean Distance Based Ranking

[ED 기반 유사 과거 데이터 수집 코드]

XGBoost [코드]

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	51.899	51.477	55.274	56.118	52.321	**56.540**	56.118
Weighted F1	0.580	0.579	0.611	0.614	0.583	**0.622**	0.616

Transformer [코드]

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	68.354	73.418	**74.684**	61.181	65.823	70.464	71.730
Weighted F1	0.635	0.640	**0.659**	0.633	0.636	0.672	0.662

DTW Distance Based Ranking

[DTW 기반 유사 과거 데이터 수집 코드]

XGBoost [코드]

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	51.899	53.586	52.321	53.164	52.743	53.586	**55.274**
Weighted F1	0.580	0.596	0.580	0.592	0.589	0.595	**0.604**

Transformer [코드]

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30

Accuracy (%)	68.354	73.418	**73.418**	67.932	70.464	71.308	69.620
Weighted F1	0.635	0.640	0.674	**0.683**	0.678	0.706	0.663

TS2Vec Embedding Based Ranking

[TS2Vec 비지도 학습 코드]

[TS2Vec 임베딩 기반 유사 과거 데이터 수집 코드]

XGBoost [코드]

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	51.899	56.540	**58.650**	50.211	53.586	51.899	52.743
Weighted F1	0.580	0.616	**0.635**	0.563	0.593	0.578	0.582

Transformer [코드]

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	68.354	73.418	**76.371**	61.603	65.823	70.042	73.000
Weighted F1	0.635	0.640	**0.669**	0.635	0.636	0.669	0.670

MultiModal Embedding Based Ranking

[모든 뉴스에 대해서 임베딩 계산하는 코드]

[구간별 뉴스 임베딩 계산하는 코드]

[차트 + 뉴스 혼합 임베딩 계산하는 코드]

[혼합 임베딩 기반 유사 과거 데이터 수집 코드]

XGBoost [코드]

	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	51.899	57.806	51.899	55.274	56.540	**57.384**	55.696
Weighted F1	0.580	0.628	0.578	0.610	0.615	**0.624**	0.609

Transformer [코드]

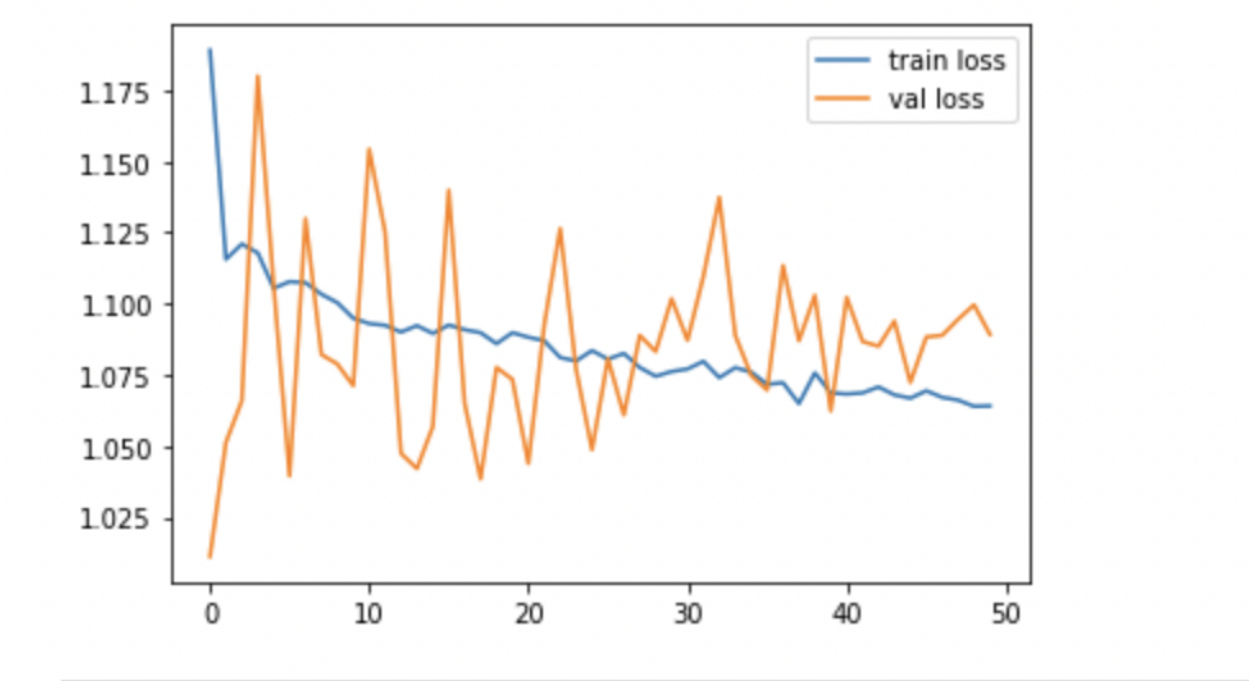
	No FE	Top 5	Top 10	Top 15	Top 20	Top 25	Top 30
Accuracy (%)	68.354	**75.949**	60.759	71.308	72.152	73.418	68.354
Weighted F1	0.635	**0.678**	0.631	0.654	0.660	0.640	0.647

분량상 트랜스포머에 관한 내용은 논문에 짧게만 언급하고, XGBoost 뿐만 아니라 다른 모델 architecture를 활용했을때도 성능 향상이 있었다고 얘기할 생각이다. 그 대신 추가적으로 XGBoost + Multimodal FE를 활용한 모델의 백테스트 성능을 추가해두면 좋을 것 같다. Live run을 진행할 시간은 없을 것 같다.

Test set에서 레이블 2가 굉장히 많다. Transformer의 경우 거의 다 2로 찍어서 Weighted F1과 accuracy가 상대적으로 높아보인다. 하지만 사실상 실제로는 쓸모없는 모델로 보인다.

전체의 80%를 candidate pool로 그리고 나머지 20%를 또 8:1:1로 쪼개서 train/val/test를 진행했는데 이 비율도 조금 조절해보면 될 것 같다.

한번 더 실험을 해볼 예정이다. 전체의 60%를 candidate pool로 그리고 나머지 40%를 또 8:1:1로 쪼개서 train/val/test로 해봤을때 성능이 올라갈까? XGBoost로만 진행해볼 예정이다. Transformer의 경우 너무 성능이 뒤죽박죽에 데이터가 작아서 학습이 조금 불안정하다. 아래는 loss graph인데



val loss가 미친듯이 왔다갔다함. LR을 좀 작게 가져가야하나…